EMJMD Engineering of Data-intensive Intelligent Software Systems (EDISS)

Research Internship Report (5 ECTS)

Out of Distribution in Object Detection Models

Supervisor: Jarno Ralli, PhD (AILiveSim)

Submitted to:
Prof. Sebastien Lafond

Submitted by:
Umar Faruk Abdullahi(2302005)

Spring 2024

Contents

1 Introduction

The code for the project is available here: https://github.com/farouqu/abo_ood

Computer vision is a interdisciplinary field that is concerned with teaching computers how to understand visual information. It is a subfield of artificial intelligence that combines techniques from computer science, mathematics, physics, and psychology to analyze and process visual data, such as images and videos. Computer vision has gained large interest and research efforts due to its applicability in various fields such as; robotics, medical imaging, surveillance and industrial automation among others[1].

The main research areas in the field of computer vision or as they are more broadly called - tasks, can be broadly categorized into three; image classification, object detection and segmentation. Image classification concerns itself with enabling machines to automatically categorize and label images based on their visual features[2] while object detection involves identifying and locating instances of visual objects of certain classes within digital images or video frames[3]. Segmentation on the other hand, involves partitioning an image into multiple meaningful segments or regions. The goal is to simplify or change the representation of an image into something that is more meaningful and easier to analyze[4].

One of the biggest challenges to deploying computer vision models specifically object detection models in the real world is the gap between the training data and real world data. Model performance in live environments may degrade due to the changes affecting the visual input features. This is due to the presence of covariate and semantic shifts that can affect the distribution of the input data. Covariate shift refers to a non-semantic change within a specific class, where the distribution of the independent variables (features) changes, but the class labels remain the same. This type of shift is often caused by changes in the environment, such as changes in lighting, camera angles, or other external factors. Conversely, semantic shift, on the other hand, represents a change in the underlying semantics or meaning of the data. This can include changes in the relationships between the input and output variables, such as changes in the way a particular feature is used to predict the output[5]. In critical scenarios, the inability of a model to accomodate these distribution shifts might lead to catastrophic circumstances.

This has lead to interest in devising methods for out-of-distribution (OOD) detection in object detection models. Out-of-distribution (OOD) detection in object detection refers to the ability of a model to identify and classify objects that are not part of the training data or are not typical of the data it was trained on. This is a crucial aspect of object detection as it ensures that the model does not misclassify or misidentify objects in real-world scenarios where the data may be different from what it was trained on. Several methods have been proposed to address OOD in computer vision including uncertainty estimation, energy based detection, confidence estimation, generative methods, ensemble methods, neighbourhood methods and threshold-based methods[6].

In this research internship, we approach the problem from a practical standpoint. We train a single-stage object detector model - YOLO based on the distribution changes

in our different datasets. We quantify the performance of our models based on the mean average precision (mAP) based on the test performance of the model in out-of-distribution (OOD) data. We curated both real world and simulated maritime environment data and converted both the required annotation required for our YOLO model. The model was trained with only real world data, only simulated data and a combination of both. To test the performance of the model on OOD data, we applied augmentations on both geometric and environmental conditions on the test data. The models performance degraded due to the augmented data being out of distribution of the data it was trained on. In the next iteration, the augmented data on both real and simulated examples was trained with that model showing greater performance for both in-distribution and out-of-distribution data.

2 Datasets

This project utilized two main maritime environment datasets. AboShips+ and Simuships.

1. **Aboships+**: The ABOships+ dataset is an improved iteration of the original ABOships dataset. It includes 9,880 images capturing maritime scenes, showcasing various types of maritime objects such as boats and cargoships. The dataset is designed to provide a comprehensive representation of maritime objects in different environments, including inshore and offshore settings[7].

Table 1. Maritime datasets for object detection from RGB imagery.

Maritime Datasets				
Denomination	**Images**	**Annotations**	**Classes**	**Resolution**
SeaShips [6]	31,455	40,077	6	1920 × 1080
SMD [11]	17,450	192,980	10	1920 × 1080
McShips [12]	14,709	26,529	13	varied
Harbour Surveillance [15]	48,966	70,513	1	2048 × 1536
SeaSAw [16]	1,900,000	14,600,000	12	varied
Maritime Ports [17]	19,337	27,849	12	1920 × 1080
ABOships [18]	9880	41,967	11	1920 × 720

Figure 1: Data distribution: Aboships+

Table 2. Properties of ABOships-PLUS: The table shows the ABOships-PLUS superclasses and their encompassing original classes of the ABOships dataset, the number of annotations (bounding boxes) for each superclass, the ratio of each superclass with respect to all annotations and the number of images that contain annotations representing each superclass.

Properties of ABOships-PLUS				
ABOships-PLUS Super Class	**ABOships Included Classes**	**Annotations per Class**	**Annotations Ratio over All Classes**	**Annotated Images per Class**
Sailboat	Sailboat	8029	24%	3756
Powerboat	Boat, Motorboat	7244	22%	4044
Ship	Passengership, Ferry, Cruiseship, Cargoship, Militaryship, Miscboat	15,272	46%	5887
Stationary	Seamark Miscellaneous floaters	2682	8%	2151

Figure 2: Distribution of vehicle types

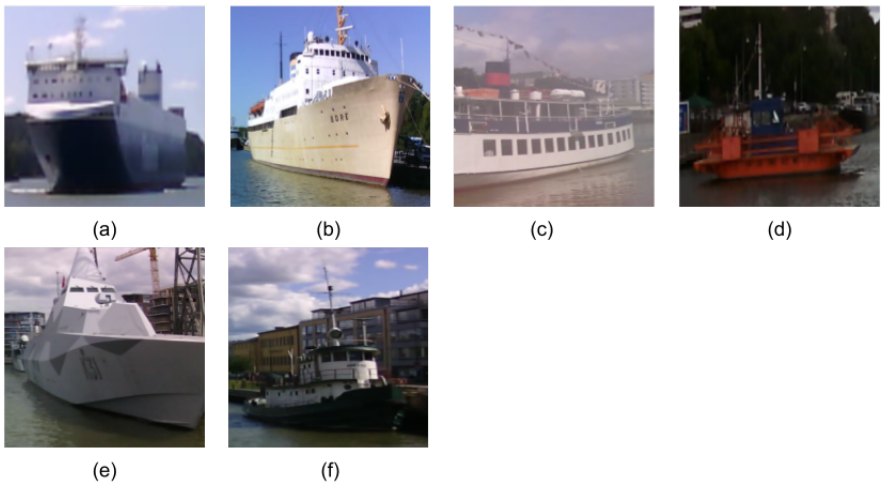

Figure 3: Sample of ships from dataset

2. **Simuships**: The SimuShips dataset is a high-resolution, simulation-based dataset for maritime environments. It consists of 9,471 high-resolution images (1920x1080) that resemble real-world maritime scenes. The dataset was acquired using the AILiveSim 3D simulation platform, which provides a realistic 3D model of the route from Turku to Ruisalo in South-West Finland. The images include a wide range of obstacle types, atmospheric conditions, and illumination variations, making it suitable for ship detection tasks. The dataset is publicly available and includes precise annotations for object detection[8].

Total number of images	6756 + 2715 = 9471
Images (without weather augmentation)	2715
Range of number of objects in images	[1, 10]
Categories	Dynamic (0), Static (1)

Figure 4: Data distribution: Simuships

Figure 5: Sample of dataset

3 Selected Approach and Tasks

In this section, the performed step-by-step tasks employed to solve the problem are highlighted. The diagram below shows a brief summary of the selected methodology:

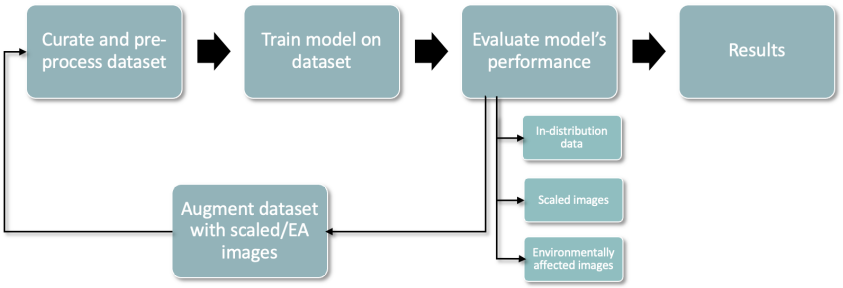

Figure 6: Selected Methodology

3.1 Model Selection

The first step of the project involved selecting the computer vision model that will be used for the project. The choice of model was between using a two-stage detector or a single stage detector. Two-stage object detectors involve a two-step process: first, they generate region proposals using a Region Proposal Network (RPN), and then they classify and refine these proposals using a convolutional neural network (CNN). This approach typically achieves higher accuracy but is slower due to the additional processing step. Examples of two-stage detectors include R-CNN and Faster R-CNN. On the other hand, one-stage object detectors directly predict bounding boxes and class probabilities without generating region proposals. They are faster and more efficient but often less accurate. The single-stage YOLO model was chosen for the task due to its speed in real world applications such as in maritime environments. YOLO (You Only Look Once) is a state-of-the-art object detection model developed by Ultralytics. It is an anchor-free model that predicts directly the center of an object instead of the offset from a known anchor box[9]. The YOLO model architecture is shown below:

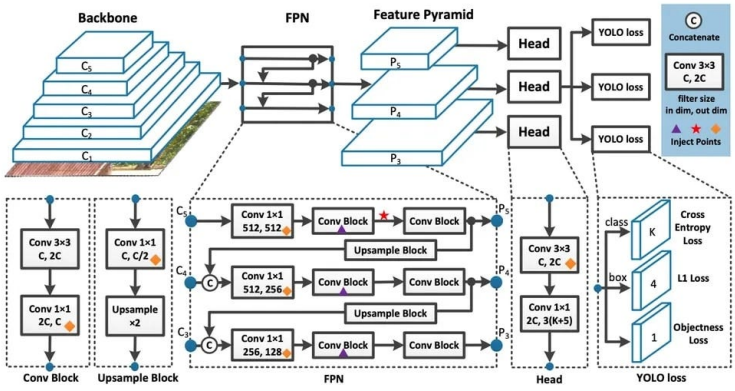

Figure 7: YOLO Architecture

3.2 Data Processing and Formatting

In the next step, the datasets were converted to the YOLO format. Since the Simuships dataset already contained YOLO format annotations, no changes were made in this step.

For the Aboships+ dataset, a custom script was run to create respective .txt files for every image based on the YOLO format. Also, Aboships+ dataset specified a different class name for every vehicle type unlike Simuships that only specified two labes: static (1) to account for all non-movable objects such as rocks, buoys and dynamic (0) for all movable objects including ships. The Aboships labels were also converted to the binary 0 and 1 representation for training the model with the combined dataset.

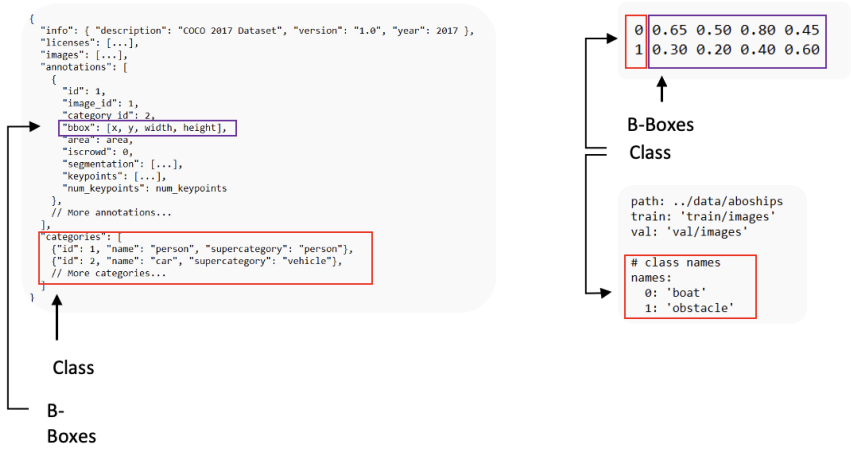

Figure 8: COCO to YOLO

Algorithm 1 Convert COCO Dataset to YOLO Format

Input: COCO_JSON_Path, YOLO_Output_Dir
Output: YOLO formatted annotation files
function ConvertCOCOToYOLO(COCO_JSON_Path, YOLO_Output_Dir)
    Load COCO_Data from COCO_JSON_Path
    Create dictionaries *Images* and *Categories* from COCO_Data
    if YOLO_Output_Dir does not exist **then**
        Create YOLO_Output_Dir
    end if
    for all Annotation in COCO_Data['annotations'] **do**
        Image_Info ← Images[Annotation['image_id']]
        $x_{min}, y_{min}, w, h \leftarrow$ Annotation['bbox']
        $x_{center} \leftarrow (x_{min} + w/2)/Image\_Info['width']$
        $y_{center} \leftarrow (y_{min} + h/2)/Image\_Info['height']$
        $w \leftarrow w/Image\_Info['width']$
        $h \leftarrow h/Image\_Info['height']$
        Class_ID ← Annotation['category_id']
        YOLO_Label_Path ← Replace extension of Image_Info['file_name'] with '.txt'
        Append $(Class\_ID, x_{center}, y_{center}, w, h)$ to YOLO_Label_Path
    end for
end function

In the second stage of the experiment, both datasets were augmented with geometric

and environmental changes to test the performance of the already trained models on out-of-distribution data and also train newer versions of the models that could be more robust to covariate shift and handle OOD data better.

3.3 Model Training and Evaluation

After processing the data appropriately, we trained various iterations of the model using the official Ultralytics Python library built on PyTorch. For all of the models, we trained on the the already pre-trained YOLO model for transfer learning. This is to use the learnt features of the YOLO model to increase the performance of our model while specializing on the maritime environment dataset. The following models were trained using the same hyperparamters shown in Table 1:

Hyperparamter	**Value**
Epochs	100
Batch Size	16
Optimizer	SGD
Learning Rate	1e-2
Confidence Threshold	0.5

Table 1: Training hyperparameters

1. **On Simulated Images**: The model was initially trained on the simulated images with no augmentation. Subsequently, it was evaluated with both simulated images and images with augmentation. Furthermore, the next iteration of the model was trained on the simulated images with augmentation and evaluated as previous.

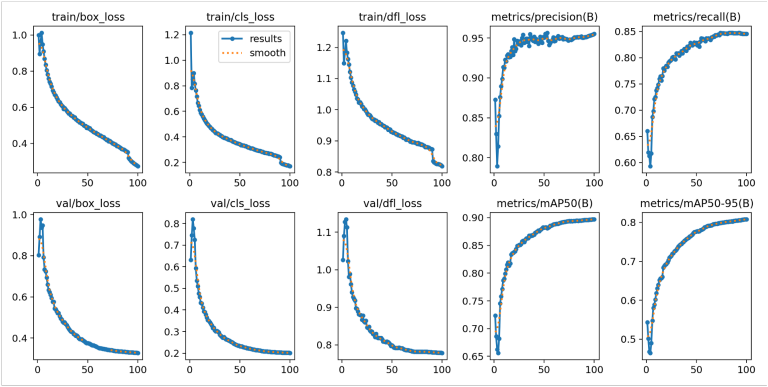

Figure 9: Training Results

2. **On Combined Images**: To create a more robust model to covariate shifts, both datasets were combined. This exposes the model to more variations in the visual representation of the maritime environment. As with the previous models, it was also evaluated on both augmented and non-augmented data.

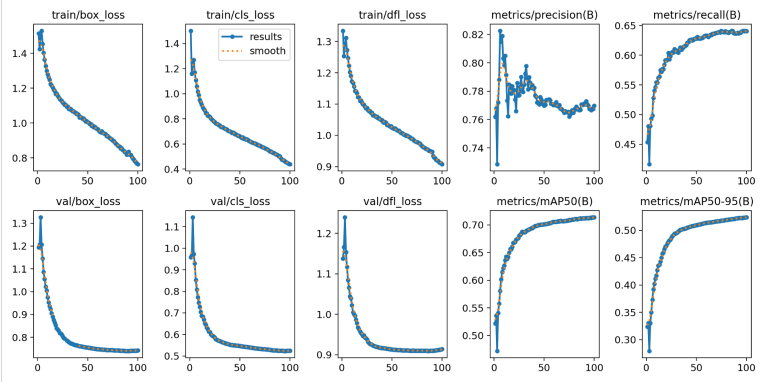

Figure 10: Training Results

3.4 Results and Discussion

Before presenting the results, the evaluation metrics used for the results are presented below. Since object detection involves classifying the detected objects to their respective classes, the precision and recall classification metrices are also used:

1. Precision: Precision is the ratio of correctly predicted positive observations to the total predicted positives. It is calculated as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

   where TP is the number of true positives and FP is the number of false positives. Precision indicates the accuracy of the positive predictions.

2. Recall: Recall, also known as sensitivity, is the ratio of correctly predicted positive observations to all the actual positives. It is calculated as:

$$\text{Recall} = \frac{TP}{TP + FN}$$

   where TP is the number of true positives and FN is the number of false negatives. Recall measures the ability to find all the relevant cases within a dataset.

3. mAP@50: Mean Average Precision at IoU 0.50 (mAP@50) is the mean of average precision values calculated at a single Intersection over Union (IoU) threshold of 0.50. It is a common metric for evaluating object detection models. It is calculated as the area under the precision-recall curve at IoU = 0.50.

4. mAP@50-95: Mean Average Precision at IoU thresholds ranging from 0.50 to 0.95 (mAP@50-95) is a more comprehensive metric that averages the precision values over multiple IoU thresholds (typically in increments of 0.05). It provides a better assessment of the detection performance across different levels of overlap.

From the results shown in Table 2, we can see the model's performance degrade significantly with an mAP drop of 58%. The model's poor performance is due to it's failure to generalize to out-of-distribution images (augmentations) that were not part of the

Test Case	Precision	Recall	mAP@50	mAP@50-95
Simulated	96%	85%	92%	86%
Simulated (+ custom augmentations)	66%	30%	34%	27%

Table 2: Results on model trained with no augmented images (Simulated images)

training data. In the subsequent iteration, the model was trained on both simulated images while adding custom augmentations. As expected, the model's performance remained almost on par with the base case when tested on the augmented data as shown in Table 3.

Test Case	Precision	Recall	mAP@50	mAP@50-95
Simulated	96%	85%	92%	86%
Simulated (+ custom augmentations)	95%	84%	89%	76%

Table 3: Results on model trained with augmented images (Simulated images)

To further increase the robustness of our model, in the next iteration, both datasets were combined to create a more diverse dataset with representation of varying vehicles and environmental conditions. The model was subsequently evaluated in a similar method as previous. The results are displayed in the tables below.

Test Case	Precision	Recall	mAP@50	mAP@50-95
Images	59%	64%	65%	47%
Images (+ custom augmentations)	40%	19%	16%	10%

Table 4: Results on model trained with no augmented images

Test Case	Precision	Recall	mAP@50	mAP@50-95
Images	66%	61%	66%	48%
Images (+ custom augmentations)	77%	58%	65%	43%

Table 5: Results on model trained with augmented images

One striking thing to note is the degraded performance of the model when trained on both the simulated and real images. This may be due to the difference in the distribution of the dataset and hence, making generalization across two diverse data distributions challenging for the object detection model. Furthermore, we faced difficulty in trying to align the annotations of the Aboships+ dataset while converting from COCO annotations to the YOLO format. The inaccuracy of the bounding boxes would lead to an increased error in the model loss and hence, negatively affect the classification score. After further research, we understood that the misalignment of annotations was based on the original COCO annotations provided in the dataset and the authors have released a newer version with corrected annotations.

In summary, our experimental findings demonstrate the efficacy of enriching the diversity of training data and incorporating variations in environmental conditions through

augmentation techniques. This augmentation-driven approach significantly enhances the performance of object detection models, rendering them more resilient and adaptable to out-of-distribution scenarios.

4 Reflection, Conclusion and Recommendations

During the course of the research internship, the most important lesson I learned was the intricate and iterative nature of scientific work. Initially, this involved understanding the existing literature within the field, evaluating prior studies, and identifying the research that was most pertinent to addressing the problem from an industry perspective.

Effective communication and diligent reporting were also key components of the internship. Regular weekly meetings with my supervisor provided a structured method to discuss progress, share our findings, and receive valuable feedback. This iterative process of communication not only facilitated a smoother project development but also improved my ability to convey complex ideas succinctly and clearly.

As a student with a keen interest in computer vision, the internship was broadened my understanding of the field. Focusing on object detection, which is one of the main tasks in the field, I was exposed to a variety of techniques and challenges that underscore the discipline. One of the most important insight I gained was the paramount importance of data distribution in training models. I learned that the diversity and representativeness of the training data are critical determinants of a model's performance in real-world applications. This understanding has profoundly influenced my appreciation of data-centric approaches in machine learning and reinforced the necessity of robust data preparation and augmentation strategies.

5 References

[1] Victor Wiley and Thomas Lucas. "Computer vision and image processing: A paper review". en. In: *Int. J. Artif. Intell. Res.* 2.1 (June 2018), p. 22.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet classification with deep convolutional neural networks". en. In: *Commun. ACM* 60.6 (May 2017), pp. 84–90.

[3] Ayoub Benali Amjoud and Mustapha Amrouch. "Object detection using deep learning, CNNs and vision transformers: A review". In: *IEEE Access* 11 (2023), pp. 35479–35516.

[4] Shervin Minaee et al. "Image segmentation using deep learning: A survey". In: (Jan. 2020). arXiv: 2001.05566 [cs.CV].

[5] Junjiao Tian et al. "Exploring covariate and concept shift for detection and calibration of out-of-distribution data". In: (Oct. 2021). arXiv: 2110.15231 [cs.LG].

[6] Jingkang Yang et al. "Generalized out-of-distribution detection: A survey". In: (Oct. 2021). arXiv: 2110.11334 [cs.CV].

[7] Bogdan Iancu et al. "A benchmark for maritime object detection with Centernet on an improved dataset, ABOships-PLUS". en. In: *J. Mar. Sci. Eng.* 11.9 (Aug. 2023), p. 1638.

[8] Minahil Raza et al. "SimuShips - A high resolution simulation dataset for ship detection with precise annotations". In: *OCEANS 2022, Hampton Roads*. Hampton Roads, VA, USA: IEEE, Oct. 2022, pp. 1–5.

[9] Ultralytics. *Home*. en. https://docs.ultralytics.com/. Accessed: 2024-6-7. Nov. 2023.